TÉCNICAS DE PROGRAMACIÓN

OBJECT ORIENTE PROGRAMING

SCOPE

ALCANCE DE VARIABLES

L.E.G.B.

LOCAL, ENCLOSING, GLOBAL, BUILT IN

```
x = 'Global x'

def outer():
    print(x)

outer()
```

GLOBAL

```
# 'Global x'
```

```
def outer(z):
    print(z)

outer('local z')
print(z)

# Error
```

LOCAL

```
x = 'Global x'

def outer():
    x = 'Local x'
    print(x)

x = 'Global x2'
outer()
```

GLOBAL

LOCAL

```
# 'Local x'
```

```
import builtins
print(dir(builtins))
```

BUILT IN

```
def outer(z):
    print(min(z))

outer([1,2,3,4,5])
```

GLOBAL

```
# 1
```

```
import builtins
print(dir(builtins))
```

BUILT IN

```
def min(iterable):
    return None

def outer(z):
    print(min(z))

outer([1,2,3,4,5])
```

GLOBAL

```
# None
```

```
def outer():
    x = 'outer x'

    def inner():
        x = 'inner x'
        print(x)

    inner()
    print(x)

outer()

# 'inner x'
# 'outer x'
```

GLOBAL

LOCAL 1

LOCAL 2

```
x = 'Global x'

def outer():
    x = 'outer x'

    def inner():
        # x = 'inner x'
        print(x)

    inner()
    print(x)

outer()

# 'outer x'
# 'outer x'
```

GLOBAL

LOCAL 1

LOCAL 2

```
x = 'Global x'

def outer():
    x = 'outer x'

    def inner():
        global x
        print(x)

    inner()
    print(x)

outer()

# 'Global x'
# 'outer x'
```

GLOBAL

LOCAL 1

LOCAL 2

CADA LENGUAJE DE PROGRAMACIÓN TIENE SU PROPIAS REGLAS DE SCOPING

ES UNA MALA PRACTICA UTILIZAR VARIABLES GLOBALES

LA CONSTANTE DE UNO PUEDE SER LA VARIABLE DE OTRO

FIRST CLASS FUNCTION

SE DICE QUE UN LENGUAJE DE PROGRAMACIÓN TIENE **FUNCIONES DE PRIMERA CLASE** SI TRATA LAS FUNCIONES COMO CIUDADANOS DE PRIMERA CLASE.

EN CONCRETO, ESTO SIGNIFICA QUE EL LENGUAJE SOPORTA TRATAR A LAS FUNCIONES COMO TYPO DE DATO

CLOSURES

ES UN METODO/TECNICA QUE PERMITE ALMACENAR/ ENCAPSULAR INFORMACIÓN DENTRO DE UNA FUNCIÓN COMO RETORNO DE OTRA

POO

OBJECT ORIENTED PROGRAMING

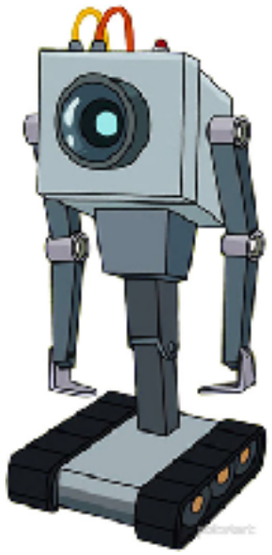
OOP ES UN PARADIGMA DE PROGRAMACIÓN, DONDE SE BUSCA RESOLVER UN PROBLEMA A TRAVÉS DE OBJETOS. LOS CUALES BUSCAN DIVIDIR UNA ÚNICA SOLUCIÓN COMPLEJA EN MÚLTIPLES SOLUCIONES MENOS COMPLEJAS QUE EN CONJUNTO LOGRAN EL MISMO OBJETIVO.

OBJETOS

LOS OBJETOS SON CONSIDERADOS ABSTRACCIONES/ MODELOS/REPRESENTACIONES ÚTILES QUE CUMPLEN UNA FUNCIÓN O RESPONSABILIDAD ESPECÍFICA.

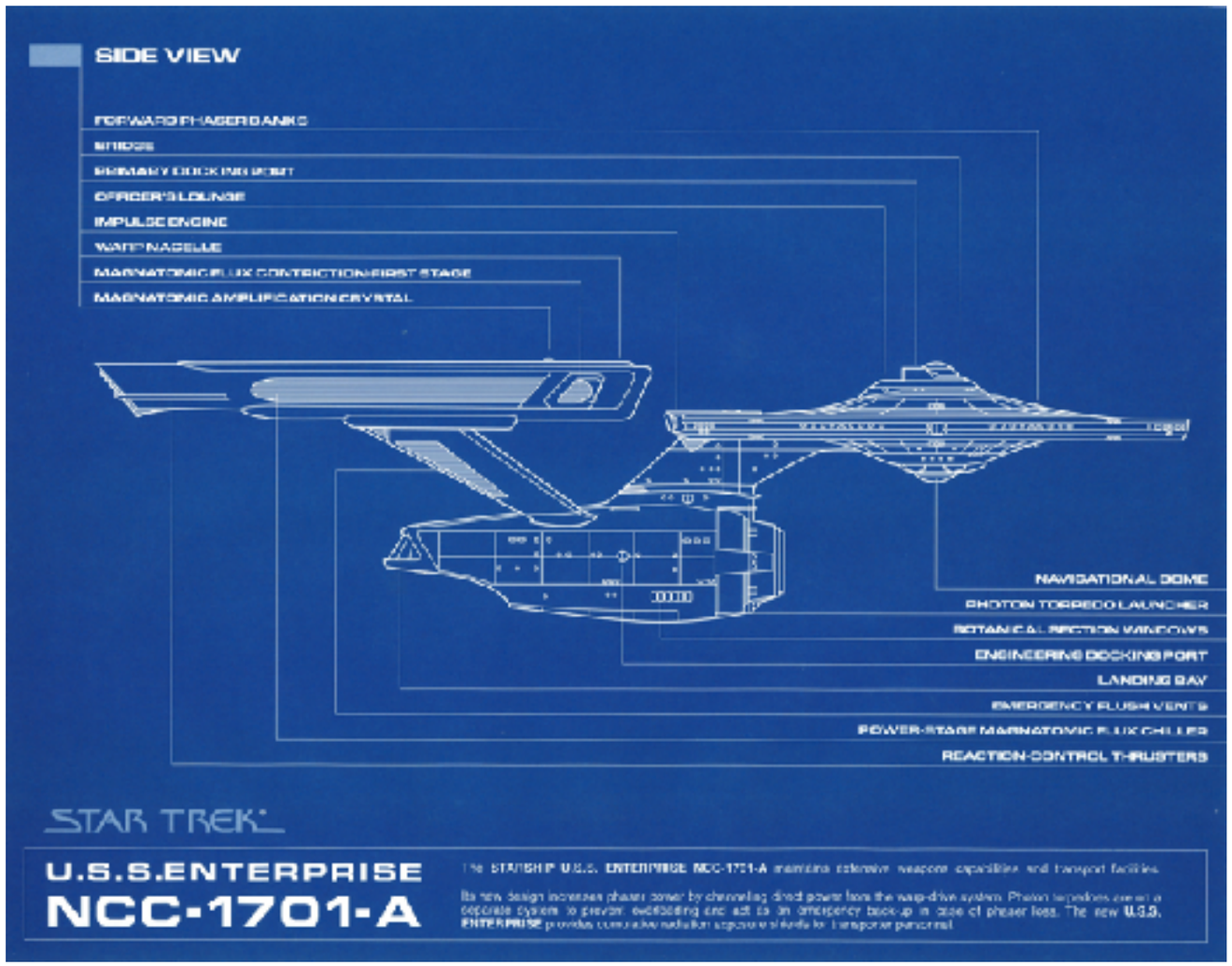
ATRIBUTOS Y METODOS

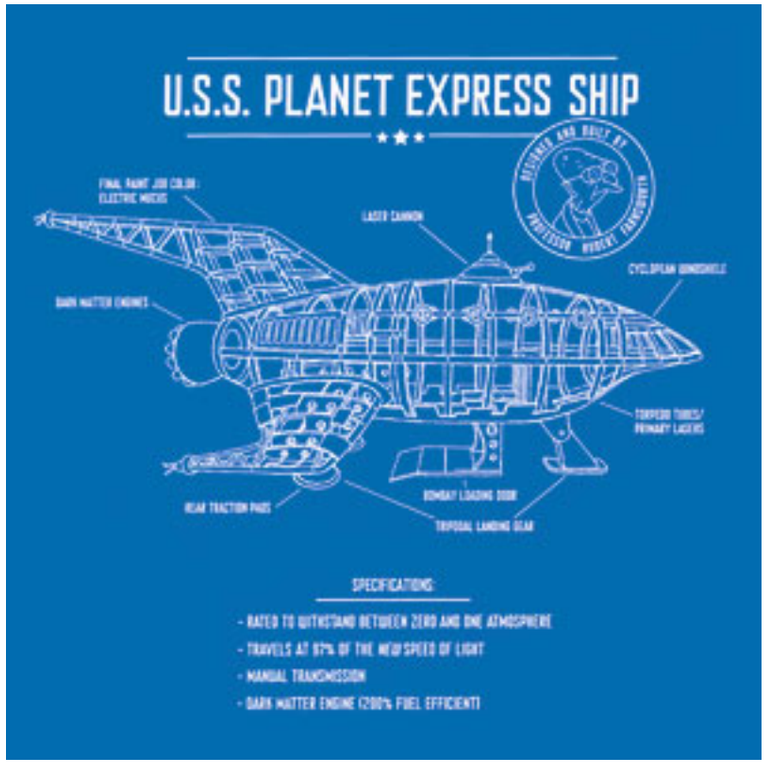
CARACTERÍSTICAS Y ACCIONES



CLASES

PLANOS DE CONSTRUCCIÓN PARA OBJETOS





INSTANCIA

ES LA REPRESENTACIÓN EN MEMORIA DE UN OBJETO, CONSTRUIDO A PARTIR DE UNA CLASE ESPECÍFICA.

SE ASIGNA A UNA VARIABLE DE REFERENCIA QUE SE UTILIZA PARA ACCEDER A TODAS LAS PROPIEDADES Y MÉTODOS DE LA INSTANCIA.

OBJETOS VS CLASES

CLASES EN PYTHON

```
class ClassName():
    pass
```

clases

- LOS NOMBRES DE CLASE DEBEN SEGUIR LA CONVENCIÓN
  `UpperCaseCamelCase`

- SIN EMBARGO LAS CLASES INCORPORADAS DE PYTHON, SON TÍPICAMENTE EN MINÚSCULAS

- LAS CLASES DE EXCEPCIÓN DEBEN TERMINAR EN "ERROR"

- LAS CLASES SOLO DEBEN DE TENER UNA Y SOLO UNA RESPONSABILIDAD*

ATRIBUTOS DE CLASE

HACEN REFERENCIA A CARACTERÍSTICAS QUE PERTENECE A LA ABSTRACCIÓN NO A LA CONCRECIÓN

Atributos de clase

MÉTODOS DE CLASE Y ESTÁTICOS

HACEN REFERENCIA A LÓGICA QUE PERTENECE A LA ABSTRACCIÓN NO A LA CONCRECIÓN

métodos de clase

GETTERS AND SETTERS

FUNCIONES O PROPIEDADES QUE EJECUTAN UNA ACCIÓN ANTES DE RETORNARSE Y DESPUÉS DE ACTUALIZARSE

OBJECT ORIENTED PROGRAMING

ENCAPSULAMINETO

LA HABILIDAD PARA OCULTAR VALORES/ESTADOS DE UN OBJETO DENTRO DE UNA CLASE, EVITANDO EL ACCESO DIRECTO ENTIDADES NO AUTORIZADAS A ELLOS.

- PRIVADO: ACCESIBLES SOLO PARA USO INTERNO DE LA CLASE
- PUBLICO: ACCESIBLE PARA TODOS

- TODAS LAS VARIABLES SE DEBEN DE COLOCAR POR DEFAULT EN PRIVADO
- ANTES DE DISPONER/USAR DE UNA CLASE HACER PÚBLICOS SOLO AQUELLOS MÉTODOS Y ATRIBUTOS NECESARIOS
- SI POSTERIORMENTE SE REQUIERE DE HACER PUBLICO UN MÉTODO O ATRIBUTO UTILIZAR GETTERS Y SETTERS
- JAMAS USAR VARIABLES GLOBALES DENTRO DE CLASES

HERENCIA

ES UN MECANISMO DONDE UN OBJETO O CLASE SE BASA EN OTRO OBJETO (HERENCIA POR PROTOTIPOS) O CLASE (HERENCIA BASADA EN LA CLASE) PARA UTILIZAR LA IMPLEMENTACIÓN BASE, ADQUIRIENDO TODAS LAS PROPIEDADES Y COMPORTAMIENTOS DEL OBJETO PADRE.

herencia

- LA HERENCIA SOLO SE UTILIZA PARA ESPECIALIZAR OBJETOS
- LA HERENCIA NO SE USA PARA COMPARTIR CÓDIGO
- DEPENDE DE LAS ABSTRACCIONES Y NO DE DE LAS CONCRECIONES
- PREFIERE COMPOSICIÓN SOBRE HERENCIA*

https://docs.python.org/3/reference/datamodel.html#special-method-names

MAGIC FUNCTIONS

FUNCIONES QUE HEREDAN TODOS LOS OBJETOS DE PYTHON DEL OBJETO BASE, APORTAN FUNCIONALIDAD DEL LEGUAJE

magic functions

POLIMORFISMO

ES LA PROVISIÓN DE UNA ÚNICA INTERFAZ A ENTIDADES DE DIFERENTES TIPOS.

RECORDAR EL PRINCIPIO DUCK TYPING

DUCK TYPING
PATTERN

SI SE ESCUCHA COMO UN PATO Y
CAMINA COMO UN PATO, ES UN PATO

EAFP PATTERN

EASIER TO ASK FOR FORGIVENESS THAN PERMISSION

¿ES JAVASCRIPT UN LEGUAJE ORIENTADO A OBJETOS?

- ¿NO TIENE CLASES O SI?
- ¿NO TIENE HERENCIA O SI?
- ¿NO TIENE ENCAPSULAMIENTO O SI?
- ¿NO TIENE POLIMORFISMO O SI?

- JS CUENTA CON OBJETOS
- ESTOS OBJETOS SE BASAN EL EL CONCEPTO DE PROTOTIPOS
- POR CONSIGUIENTE "NO EXISTEN LAS CASES"
- PERO EN 2016 EN ECMASCRIPT 6 SE INTRODUCE EL KEYWORD "CLASS" ENTONCES YA HAY CLASES PERO NO
- JS TIENE HERENCIA POR PROTOTIPOS, NO ES UNA HERENCIA TRADICIONAL PERO ES HERENCIA
- JS CUMPLE AL 100% EL CONCEPTO DE POLIMORFISMO
- JS NO TIENE ENCAPSULAMIENTO

JAVASCRIPT NO ES UN LEGUAJE ORIENTADO A OBJETOS

PERO SE PUEDE USAR COMO UNO, TENIENDO LAS BUENAS PRACTICAS Y PRECAUCIONES ADECUADAS

PROTOCOLOS / INTERFACES / ABSTRACT BASE CLASS

UN CONTRATO QUE OBLIGA A UN OBJETO A IMPLEMENTAR O TENER CIERTO COMPORTAMIENTO

- PERMITE ESTANDARIZAR LA HERENCIA (EN PYTHON)
- PERMITE TENER UNA ALTERNATIVA A HERENCIA, EXTENDIENDO LA FUNCIONALIDAD (EN OTROS LENGUAJES)

DEPENDENCY INJECTION PATTERN

PATTERN

SE DEFINE COMO LA INTEGRACIÓN DE OBJETOS COMO PARTE DE LOS ATRIBUTOS DE OTRO OBJETO

dependency injection

- LOS CLIENTES QUE SON MÁS INDEPENDIENTES APOYANDO AL UNIT TESTING
- REDUCCIÓN DE CÓDIGO
- PERMITE A UN CLIENTE ELIMINAR TODO CONOCIMIENTO DE UNA IMPLEMENTACIÓN CONCRETA. ESTO AYUDA A AISLAR AL CLIENTE DEL IMPACTO DE LOS CAMBIOS Y DEFECTOS DEL DISEÑO (ACOPLAMIENTO).
- PROMUEVE LA REUTILIZACIÓN, LA TESTABILIDAD Y LA FACILIDAD DE MANTENIMIENTO.
- LA INYECCIÓN DE DEPENDENCIA PERMITE A UN CLIENTE LA FLEXIBILIDAD DE SER CONFIGURABLE

COMPOSICIÓN
PATTERN

ES UN PATRÓN DE REUTILIZACIÓN, ATRAVEZ LA IMPEMENTACION DE COMPORTAMIENTOS POR CLASES INFECTADAS EN LUGAR DE LA HERENCIA DE UNA CLASE DE PADRE.

COMPLETA LA CLASE `Gun` REPRESENTA UNA PISTOLA, ESTA CLASE DEBE DE CUMPLIR LOS SIGUIENTES REQUERIMIENTOS:

- EL USUARIO DEBE PODER MANIPULAR EL SEGURO DE LA PISTOLA
- EL USUARIO DEBE PODER DISPARAR, GASTANDO UNA BALA POR TIRO
- SI EL CARTUCHO ESTA BACIO EL USUARIO NO PUEDE DISPARAR
- SI EL SEGURO ESTA PUESTO EL USUARIO NO PUEDE DISPARAR
- EL USUARIO DEBE PODER CARGAR MUNICIÓN
- EL USUARIO SOLO PUEDE CARGAR TANTA MUNICIÓN COMO LO PERMITA EL CARGADOR DEL ARMA
- EL USUARIO DEBE SABER CUANTAS BALAS NO PUDO METER AL CARTUCHO
- EL CARGADOR DEL ARMA DEBE DE SER CONFIGURABLE AL MOMENTO DE CREAR EL ARMA

RETO 13: TERMOSTATO

ESTAS TRABAJANDO EN EL DISPLAY DE UN TERMOSTATO, EL CUAL TIENE UNA OPCIÓN DE VISUALIZAR ALGUNAS ESTADÍSTICAS HISTORIAS:

TEMPERATURA MÁXIMA

- TEMPERATURA MÍNIMA
- TEMPERATURA PROMEDIO

ESCRIBE UNA CLASE TRACKER CON ESTOS MÉTODOS:

- INSERT – REGISTRA UNA NUEVA TEMPERATURA

LOS SIGUIENTES ATRIBUTOS

- MAX – DEVUELVE LA TEMPERATURA MÁS ALTA DE TODOS LOS VALORES REGISTRADOS
- MIN – DEVUELVE LA TEMPERATURA MÁS BAJA DE TODOS LOS VALORES REGISTRADOS
- AVERAGE – DEVUELVE LA TEMPERATURA PROMEDIO

CONSIDERACIONES:

- TOPAS LOS ATRIBUTO AL CONSULTARLOS DEBEN DE EJECUTARSE EN O(1)
- LAS UNIDADES SON FAHRENHEIT, ASÍ QUE PODEMOS ASUMIR QUE TODAS ESTARÁN EN EL RANGO DE 0 A 110 DE 110.

IMPLEMENTA EL ALGORITMO DE LA LA PAREJA NO TAN FELIZ, PERO ESTA VEZ CON CLASES

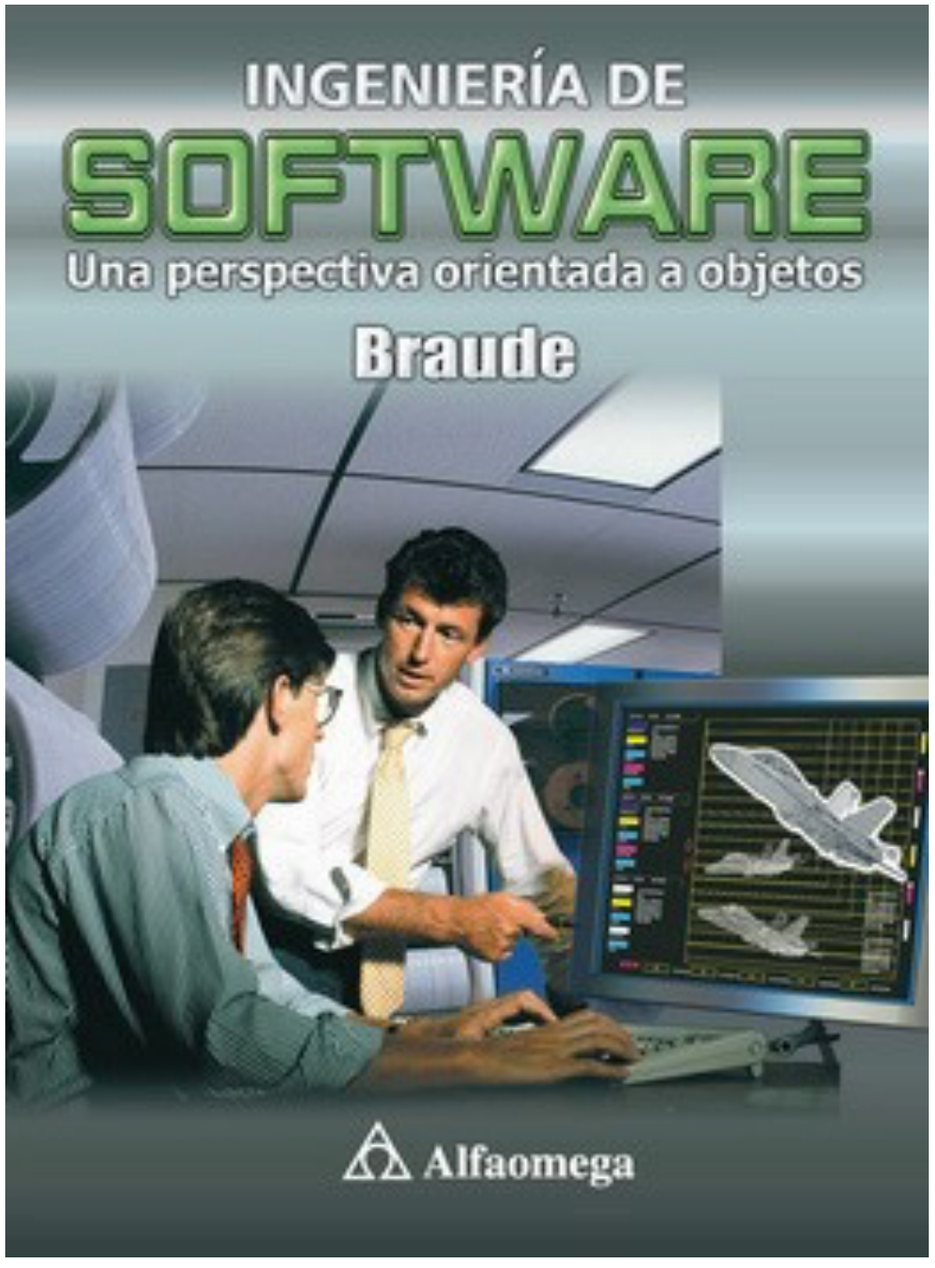
- CLASE **`PERSON`**: CONTIENES TODAS LAS PROPIEDADES Y MÉTODOS DE UNA PERSONA
- CLASE **`PARSER`**: ES RESPONSABLE DE TRASFORMAR LOS DATOS DEL JSON A INSTANCIAS DE LAS CLASE **`PERSON`**
- CLASE **`MATCHES`**: ES RESPONSABLE DE CREAR INSTANCIAS DE LA CLASE **`CUPLE`**
- CLASE **`CUPLE`**: ES RESPONSABLE DE IMPRIMIR BONITO LAS PAREJAS

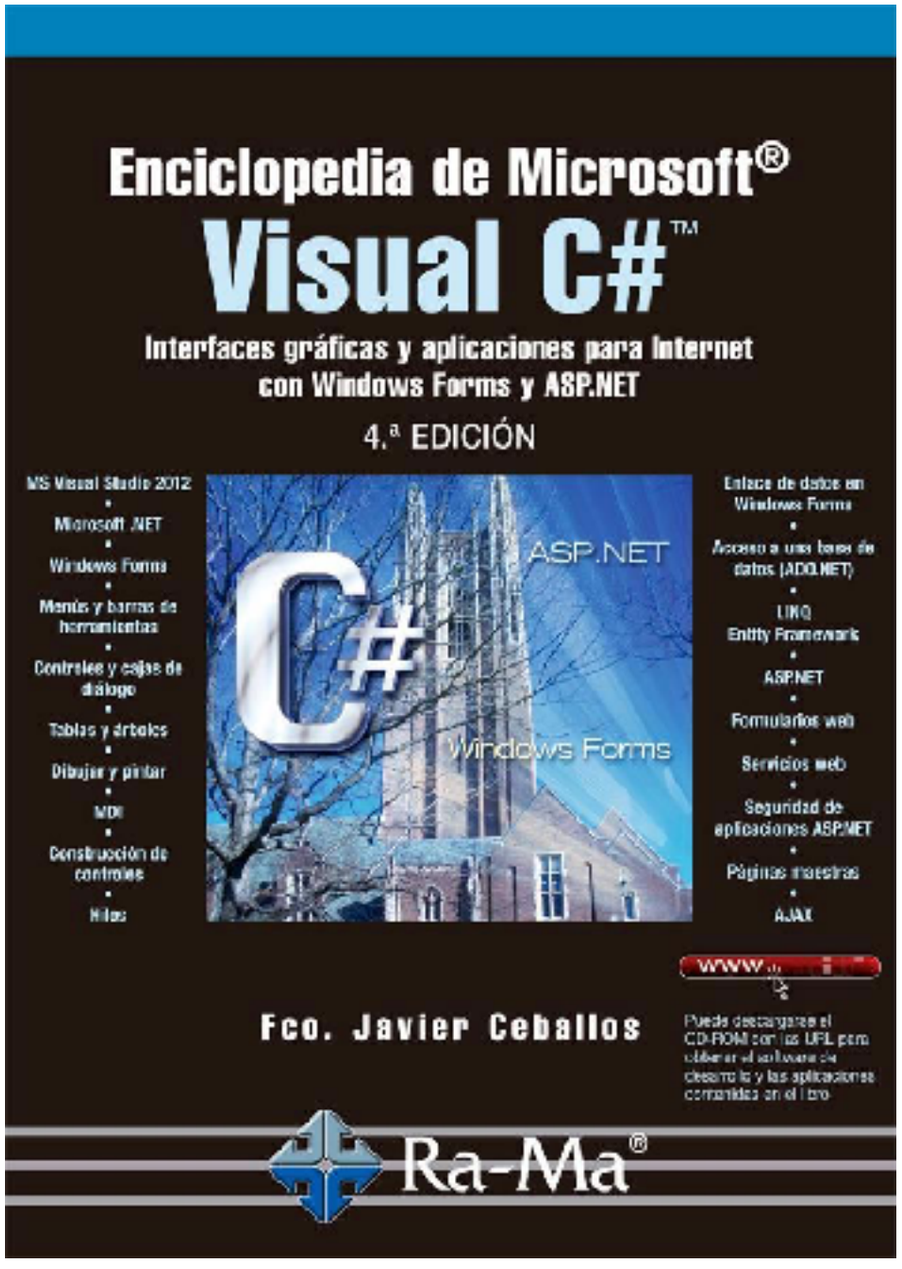
SOLO DEBES ESCRIBIR CODIGO DENTRO DE LAS CLASES MARCADAS, NADA DE CODIGO FUERA

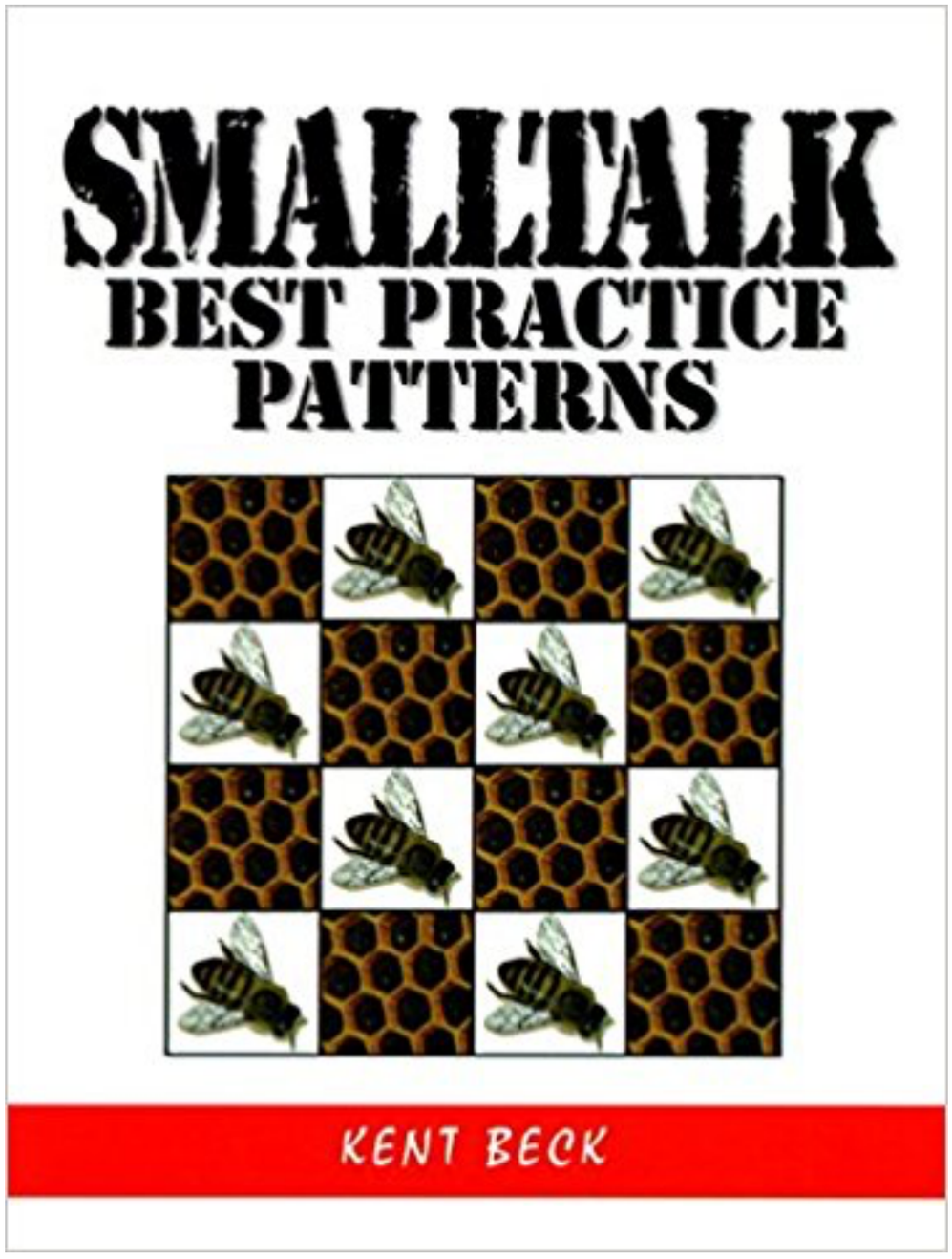
RETO 14: PAREJA NO TAN FELIZ (REFACTOR)

```
import json

class Person():
    pass

class Parser():
    pass

class Matcher():
    pass

class Cuple():
    pass

if __name__ == "__main__":
    propusers = Parser.open_json('female')
    resivers = Parser.open_json('male')
    matcher = Matcher(propusers, resivers)
    print(matcher.match())
```

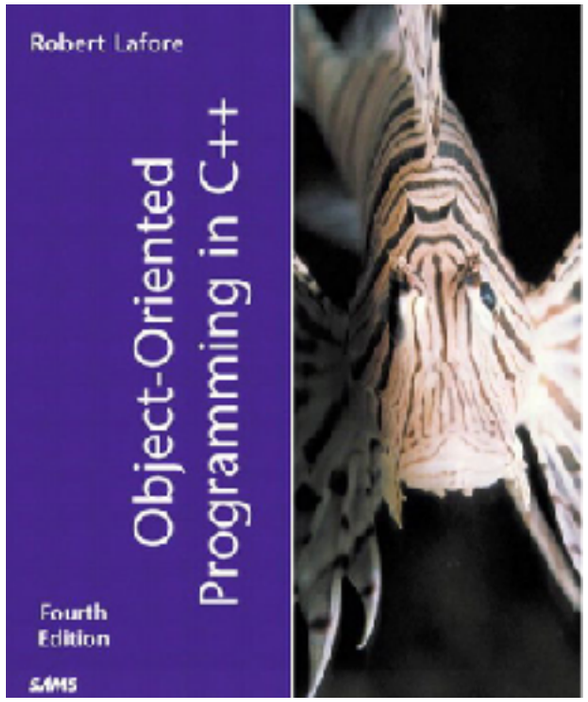
LOS FEOS

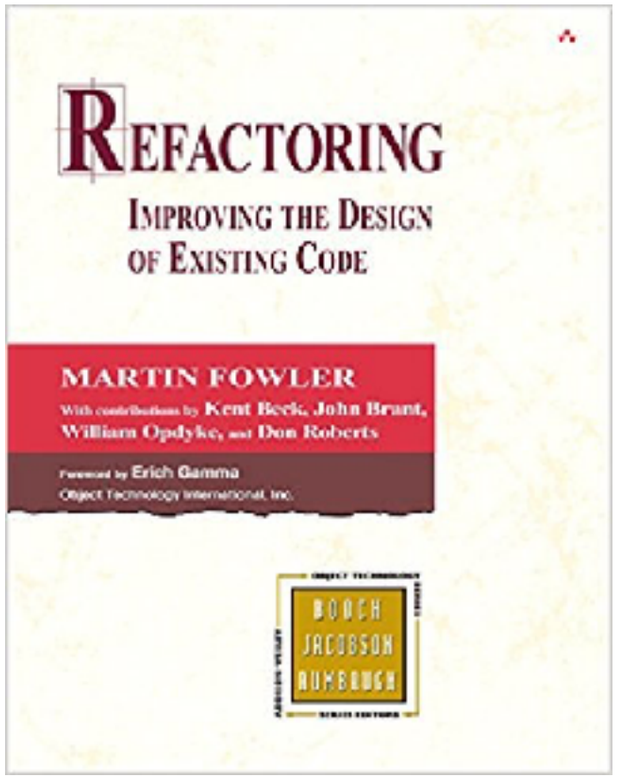

INGENIERÍA DE
SOFTWARE
Una perspectiva orientada a objetos
Braude
Alfaomega

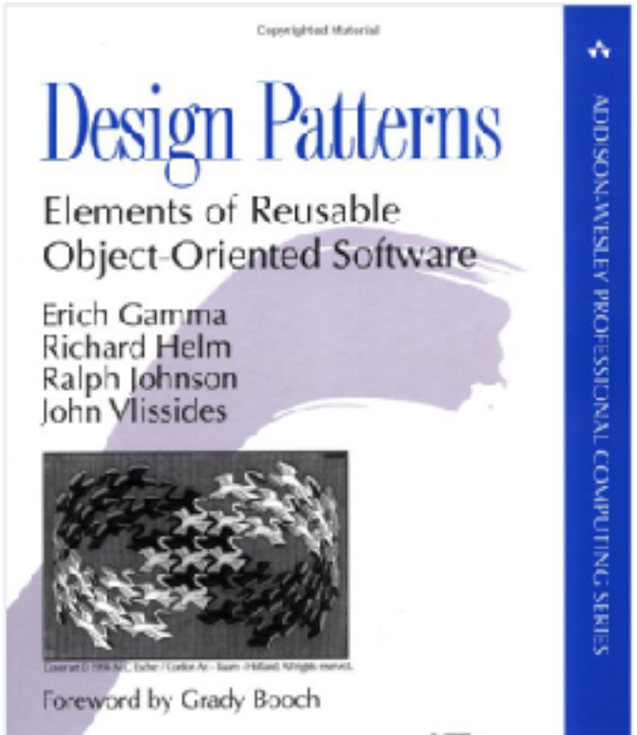

Enciclopedia de Microsoft®
Visual C#™
Interfaces gráficas y aplicaciones para Internet con Windows Forms y ASP.NET
4.ª EDICIÓN
MS Visual Studio 2012
Microsoft .NET
Windows Forms
Menús y barras de herramientas
Controles y cajas de diálogo
Tablas y árboles
Dibujar y pintar
MDI
Construcción de controles
Hilos
C#
ASP.NET
Windows Forms
Enlace de datos en Windows Forms
Acceso a una base de datos (ADO.NET)
LINQ
Entity Framework
ASP.NET
Formularios web
Servicios web
Seguridad de aplicaciones ASP.NET
Páginas maestras
AJAX
www...
Fco. Javier Ceballos
Puede descargarse el CD-ROM con las URL para obtener el software de desarrollo y las aplicaciones comentadas en el libro
Ra-Ma®

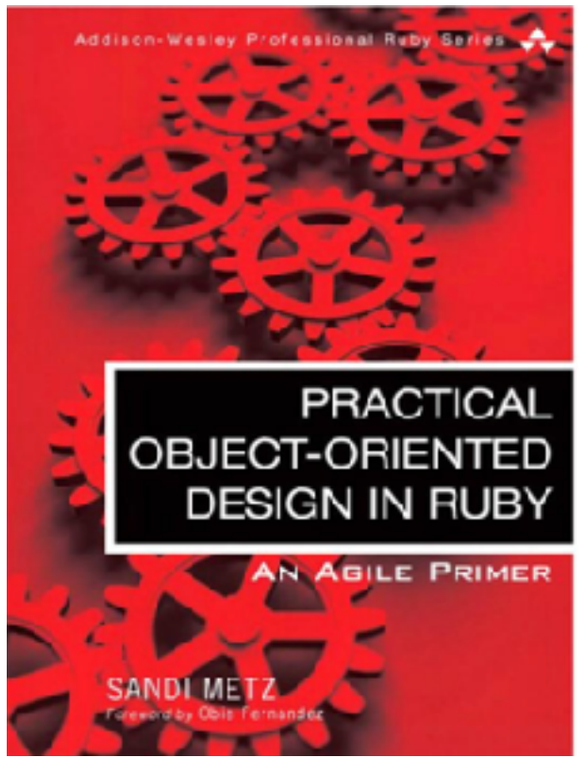

SMALLTALK
BEST PRACTICE
PATTERNS
KENT BECK

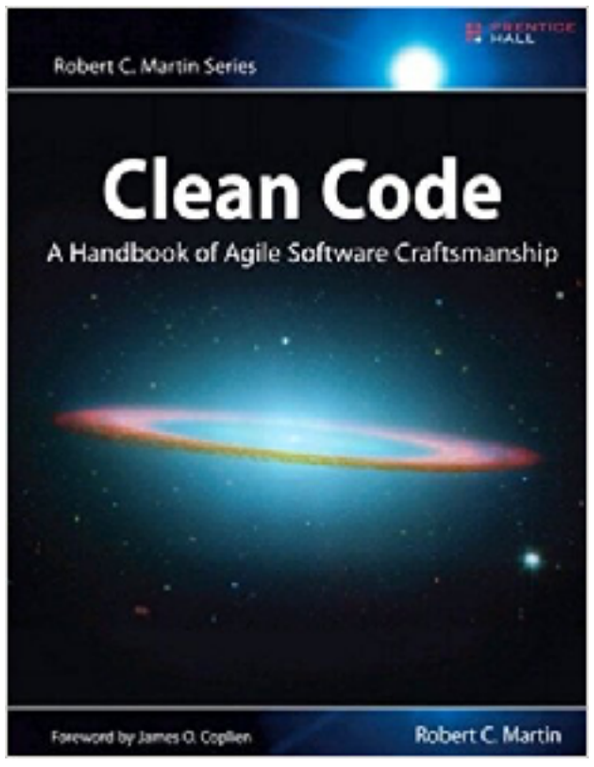
LOS GRANDES

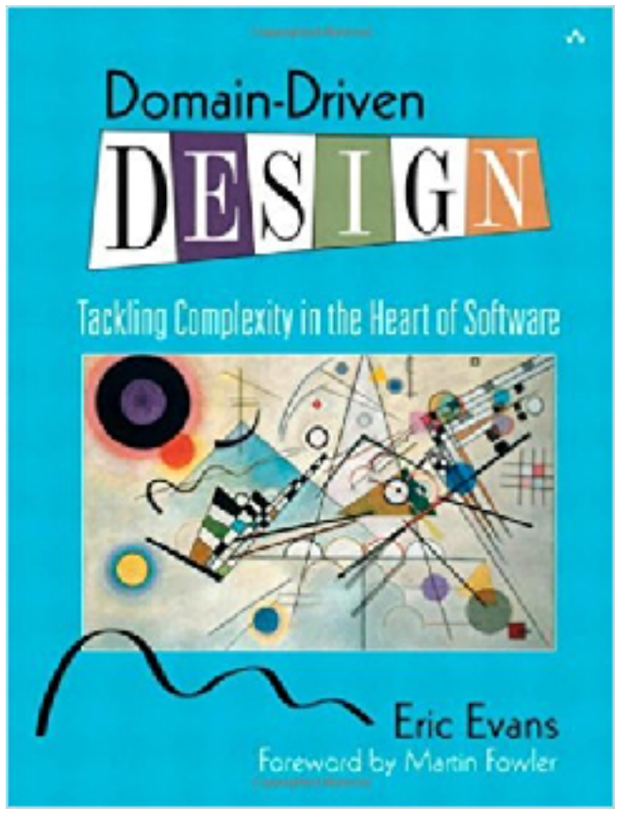

Robert Lafore
Object-Oriented Programming in C++
Fourth Edition
SAMS


REFACTORING
IMPROVING THE DESIGN OF EXISTING CODE
MARTIN FOWLER
With contributions by Kent Beck, John Brant, William Opdyke, and Don Roberts
Foreword by Erich Gamma
Object Technology International, Inc.
BOOCH
JACOBSON
RUMBAUGH

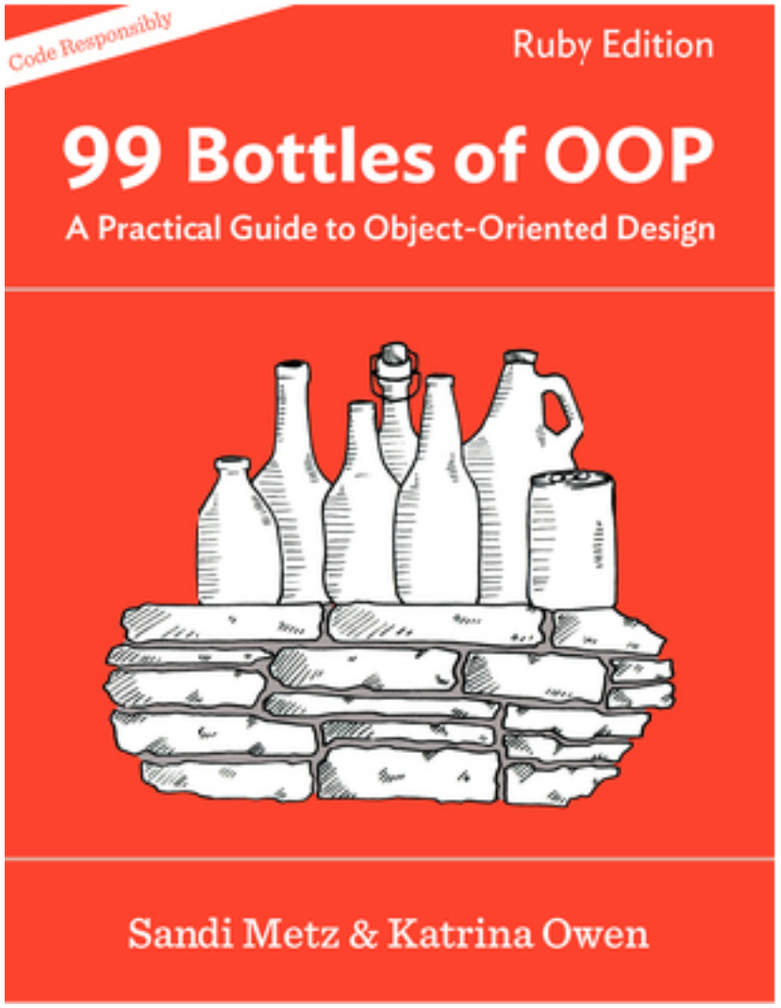

Copyrighted Material
Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES


Addison-Wesley Professional Ruby Series
PRACTICAL OBJECT-ORIENTED DESIGN IN RUBY
AN AGILE PRIMER
SANDI METZ
Foreword by Obie Fernandez


Robert C. Martin Series
PRENTICE HALL
Clean Code
A Handbook of Agile Software Craftsmanship
Foreword by James O. Coplien
Robert C. Martin


Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

COOL


Code Responsibly
Ruby Edition
99 Bottles of OOP
A Practical Guide to Object-Oriented Design
Sandi Metz & Katrina Owen


O'REILLY
10th Anniversary Updated for Java 8
Head First
Design Patterns
A Brain-Friendly Guide
Eric Freeman & Elisabeth Robson

1. SCOPING
2. CLOSURES
3. OBJETOS Y CLASES
4. ATRIBUTOS Y METODOS
5. HERENCIA, ENCAPSULAMIENTO Y POLIMORFISMO
6. DUCK TYPING PATTERN
7. EAFP PATTERN
8. DEPENDENCY INJECTION PATTERN
9. COMPOSICIÓN